|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |
| --- | --- |

ФАКУЛЬТЕТ ИУ, Информатика и системы управления

КАФЕДРА ИУ7, Программное обеспечение ЭВМ и информационные технологии

**Отчет**

***По лабораторной работе №5 (часть с Унификацией):***

***Процедура резолюции с унификацией***

Студент ИУ7-32М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Андреев**

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ З.Н. Русакова**

(Подпись, дата) (И.О.Фамилия)

*2024 г.*

# Содержание

[Содержание 4](#_Toc181966597)

[ВВЕДЕНИЕ 5](#_Toc181966598)

[Структура программы 6](#_Toc181966599)

[Работа программы 7](#_Toc181966600)

[ПРИЛОЖЕНИЕ А 8](#_Toc181966601)

# ВВЕДЕНИЕ

В данной лабораторной работе реализована процедура унификации, которая является ключевым элементом в логическом выводе и резолюции в численных логических системах. Целью работы было создание системы на языке C++, которая объединяет логические атомы с использованием унификации терминов.

# Структура программы

Программа состоит из нескольких файлов, в которых описаны основные структуры и методы для унификации:

* Файлы с типами и структурами
  + atom.h и atom.cc: определяет структуру Atom, которая представляет логический атом с именем и набором терминалов.
  + terminal.h и terminal.cc: абстрактный класс Terminal, представляющий базовый элемент (переменные и константы).
  + constant.h: наслежует класс Terminal и определяет константы.
  + variable.h: наслежует класс Terminal и определяет переменные.
* Логика унификации
  + unification.h и unification.cc: отвечает за процесс унификации двух атомов. Основная функция Unify() использует алгоритм, основанный на разрешении переменных и сопоставлении терминалов.
* Утилиты и вспомогательные модули
  + name.h: предоставляет типы для имен, используемых в терминах.
  + io\_join.h: утилита для вывода коллекций в поток с делимитером.
* Система сборки
  + Lists.txt: файл для сборки проекта с помощью CMake.
* Главный файл программы
  + main.cc: демонстрирует работу процедуры унификации. Создаются две структуры Atom, и происходит попытка их унификации.

# Основные компоненты и их функциональность

**Структуры Atom и Terminal**

* Atom содержит имя и вектор указателей на терминалы. Он используется как основной элемент унификации.
* Terminal является абстрактным классом и может представлять либо константу (Constant), либо переменную (Variable).

**Логика унификации**

Процесс унификации реализован в классе Unification, который:

* Сравнивает атомы по именам и количеству терминалов.
* Выполняет поэлементное сравнение терминалов.
* Обрабатывает случаи, когда терминалы являются переменными или константами, с соответствующими подстановками переменных.
* Учитывает связи между переменными, которые приводят к цепной замене.

# Реализация

Наиболее значимая часть этой работы была реализована в файле unification.cc. Основные шаги процесса унификации:

1. Сравнение имен атрибута: если имена атомов различаются, унификация невозможна.
2. Сравнение количества терминалов: для успешной унификации количества терминалов должны совпадать.
3. Унификация терминалов:
   1. Если терминал — переменная, она может связываться либо с другой переменной, либо с константой.
   2. Если терминал — константа, они должны совпадать, чтобы продолжить унификацию.
4. Отслеживание и разрешение связей: реализуется через наборы связей, которые объединяют переменные.

# Тестирование программы

Файл main.cc демонстрирует создание двух атомов и их унификацию. При успешной унификации выводится сообщение "SUCCESS", в противном случае — "FAILURE". Логика и результат унификации детализируется в консольном выводе, позволяя понять этапы работы алгоритма.

Для проведения тестирования программы возьмем следующие входные данным:

|  |
| --- |
| auto v1 = Variable{"v1"};  auto v2 = Variable{"v2"};  auto v3 = Variable{"v3"};  auto v4 = Variable{"v4"};  auto v5 = Variable{"v5"};  auto v6 = Variable{"v6"};  auto v7 = Variable{"v7"};  auto v8 = Variable{"v8"};  auto v9 = Variable{"v9"};  auto v10 = Variable{"v10"};  auto v11 = Variable{"v11"};  auto c1 = Constant{"1"};  auto c2 = Constant{"2"};  auto a1 = Atom{  "A", {&v1, &v3, &v4, &v5, &v1, &v1, &v7, &v9, &v7, &v10, &v8, &v11, &v4}};  auto a2 = Atom{  "A", {&v2, &v1, &c1, &v4, &v6, &v5, &v8, &v10, &v9, &v7, &v1, &c2, &v11}};  const auto res = Unify(a1, a2); |

Листинг 1 – Входные данные

|  |
| --- |
| 1. [Унификация] 2. Атом 1: A(var(v1), var(v3), var(v4), var(v5), var(v1), var(v1), var(v7), var(v9), var(v7), var(v10), var(v8), var(v11), var(v4)) 3. Атом 2: A(var(v2), var(v1), const(1), var(v4), var(v6), var(v5), var(v8), var(v10), var(v9), var(v7), var(v1), const(2), var(v11)) 4. [Пара 1] var(v1) — var(v2) 5. Переменная var(v1) встречается первый раз 6. Переменная var(v2) встречается первый раз 7. Свяжем эти переменные 8. {v1, v2}: ??? 9. [Пара 2] var(v3) — var(v1) 10. Переменная var(v3) встречается первый раз 11. Переменная var(v1) уже встречалась ранее, но на неё не распространена константа 12. Добавим var(v3) к связи var(v1) 13. {v1, v2, v3}: ??? 14. [Пара 3] var(v4) — const(1) 15. Согласуем подстановку var(v4) = const(1) 16. {v4}: 1 17. {v1, v2, v3}: ??? 18. [Пара 4] var(v5) — var(v4) 19. Переменная var(v5) встречается первый раз 20. Распространим подстановку var(v4) = const(1) и на var(v5) 21. {v4, v5}: 1 22. {v1, v2, v3}: ??? 23. [Пара 5] var(v1) — var(v6) 24. Переменная var(v1) уже встречалась ранее, но на неё не распространена константа 25. Переменная var(v6) встречается первый раз, добавим её к связи var(v1) 26. {v4, v5}: 1 27. {v1, v2, v3, v6}: ??? 28. [Пара 6] var(v1) — var(v5) 29. Переменная var(v1) уже встречалась ранее, но на неё не распространена константа 30. У переменной var(v5) согласована подстановка const(1), распространим её и на связь var(v1) 31. Согласуем подстановку v1 = const(1) 32. Согласуем подстановку v2 = const(1) 33. Согласуем подстановку v3 = const(1) 34. Согласуем подстановку v6 = const(1) 35. {v1, v2, v3, v4, v5, v6}: 1 36. [Пара 7] var(v7) — var(v8) 37. Переменная var(v7) встречается первый раз 38. Переменная var(v8) встречается первый раз 39. Свяжем эти переменные 40. {v1, v2, v3, v4, v5, v6}: 1 41. {v7, v8}: ??? 42. [Пара 8] var(v9) — var(v10) 43. Переменная var(v9) встречается первый раз 44. Переменная var(v10) встречается первый раз 45. Свяжем эти переменные 46. {v1, v2, v3, v4, v5, v6}: 1 47. {v7, v8}: ??? 48. {v10, v9}: ??? 49. [Пара 9] var(v7) — var(v9) 50. Переменная var(v7) уже встречалась ранее, но на неё не распространена константа 51. Переменная var(v9) уже встречалась ранее, но на неё не распространена константа 52. Переменные находятся в разных связях, объединим их 53. {v1, v2, v3, v4, v5, v6}: 1 54. {v10, v7, v8, v9}: ??? 55. [Пара 10] var(v10) — var(v7) 56. Переменная var(v10) уже встречалась ранее, но на неё не распространена константа 57. Переменная var(v7) уже встречалась ранее, но на неё не распространена константа 58. Переменные уже находятся в одной связи, ничего не делаем 59. {v1, v2, v3, v4, v5, v6}: 1 60. {v10, v7, v8, v9}: ??? 61. [Пара 11] var(v8) — var(v1) 62. Переменная var(v8) уже встречалась ранее, но на неё не распространена константа 63. У переменной var(v1) согласована подстановка const(1), распространим её и на связь var(v8) 64. Согласуем подстановку v10 = const(1) 65. Согласуем подстановку v7 = const(1) 66. Согласуем подстановку v8 = const(1) 67. Согласуем подстановку v9 = const(1) 68. {v1, v10, v2, v3, v4, v5, v6, v7, v8, v9}: 1 69. [Пара 12] var(v11) — const(2) 70. Согласуем подстановку var(v11) = const(2) 71. {v11}: 2 72. {v1, v10, v2, v3, v4, v5, v6, v7, v8, v9}: 1 73. [Пара 13] var(v4) — var(v11) 74. Подстановка var(v4) = const(1) 75. Подстановка var(v11) = const(2) 76. Подстановки не совпали 77. FAILURE |

Листинг 2 – Вывод программы